Java Programming
Simple JavaGUI

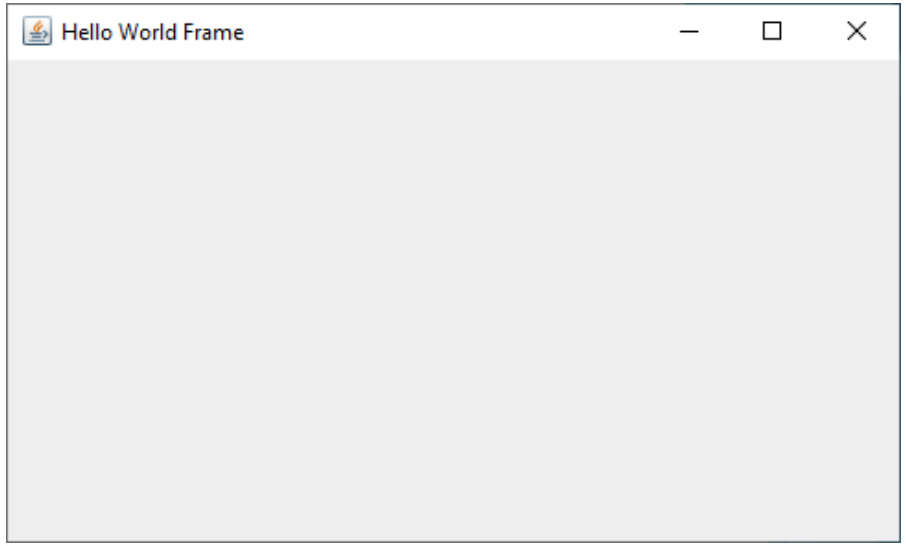
Presented By
Dr. Eman Hesham, DBA.

Java™ Education and Technology Services

Invest In Yourself , Develop Your Career

Create Simple GUI – java Swing Frame

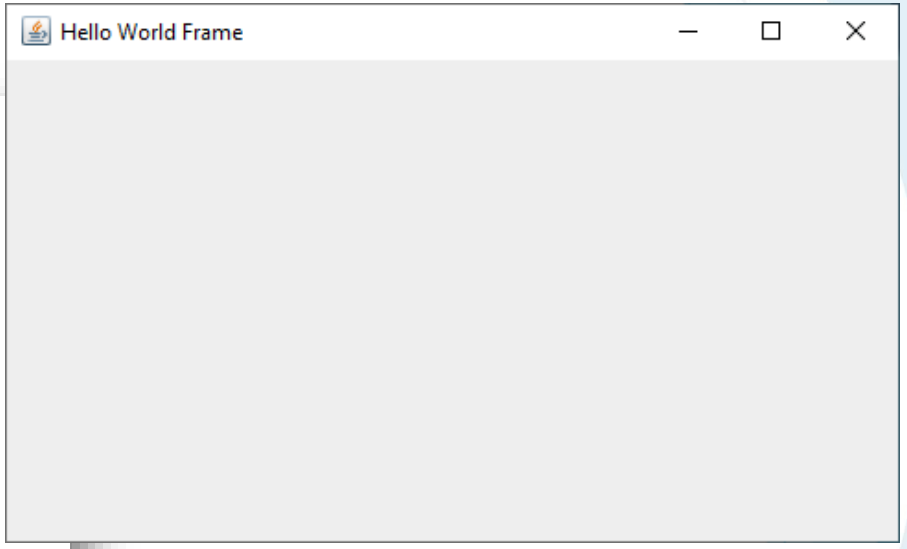
- JFrame is a class of the javax.swing package .
- This is the top-level window, with border and a title bar.
- JFrame class has various methods which can be used to customize it.



Create Simple GUI – java Swing Frame

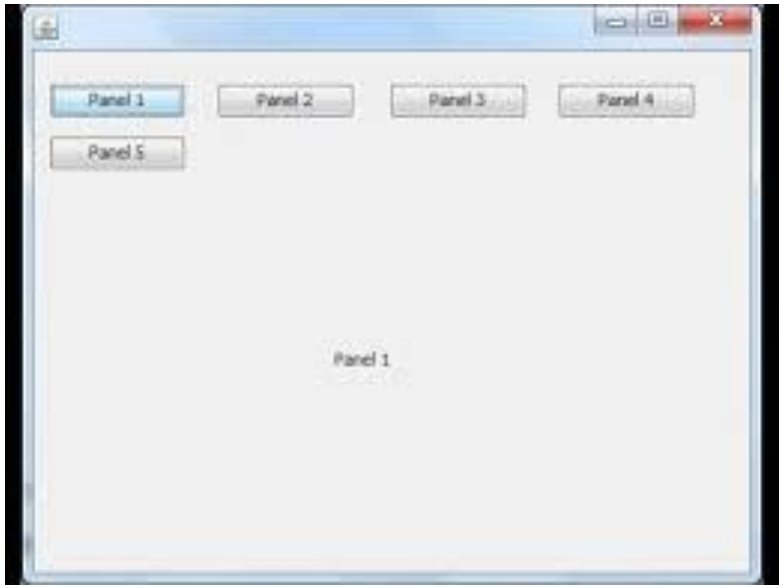
Java™ Education and Technology Services

```
package thrdPkg;

import javax.swing.JFrame;

public class SimpleFrame {

    public static void main(String[] argv) {
        // Create a frame
        JFrame f = new JFrame();

        // Set the title and other parameters.
        f.setTitle("Hello World Frame");
        f.setSize(500, 300);
        f.setVisible(true);
    }
}
```



Create Simple GUI – java Swing Panel

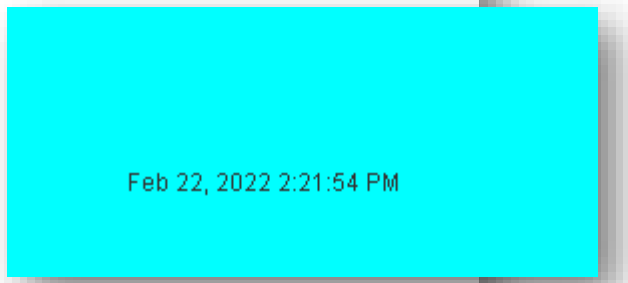
- JPanel, a part of the Java Swing package, is a container that can store a group of components.
- The main task of JPanel is to organize components, various layouts can be set in JPanel which provide better organization of components.



Create Simple GUI – java Swing Panel

```java
import java.awt.Color;
import java.awt.Graphics;
import java.util.Date;
import javax.swing.JPanel;

class MyPanel extends JPanel {

    public MyPanel() {
        this.setBackground(Color.cyan);
    }

    @Override
    public void paintComponent(Graphics g) {
        // Always call super.paintComponent (g):
        super.paintComponent(g);
        g.drawString(new Date().toLocaleString(), 100, 100);
    }
}
```

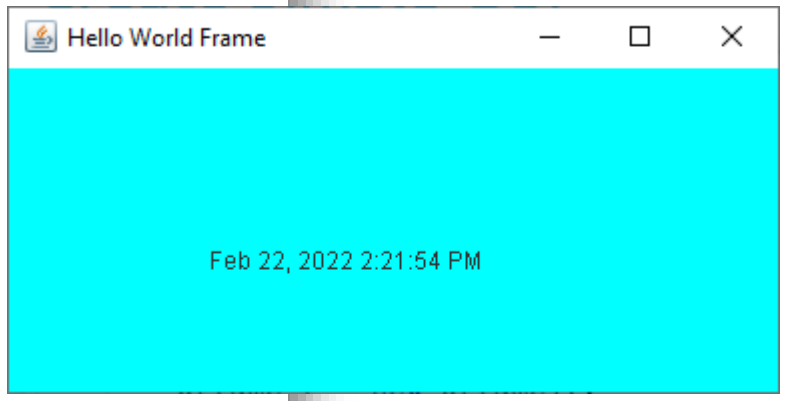
1



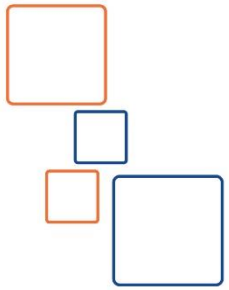
Create Simple GUI

```
package thrdPkg;

import javax.swing.JFrame;

public class SimpleFrame {

    public static void main(String[] argv) {
        // Create a frame
        JFrame f = new JFrame();

        // Set the title and other parameters.
        f.setTitle("Hello World Frame");
        MyPanel mp=new MyPanel();
        f.setContentPane(mp);
        f.setSize(400, 200);
        f.setVisible(true);
    }
}
```

2



Java Programming
Essential Java Classes (Graphics-Font-Color)



Java™ Education and Technology Services

Invest In Yourself, Develop Your Career

Graphics Class

- The Graphics object is your means of communication with the graphics display.
- You can use it to draw strings, basic shapes, and show images.
- You can also use it to specify the color and font you want.
- You can write a string using the following method:

```
void drawString(String str, int x, int y)
```

Graphics Class

- Some basic shapes can be drawn using the following methods:

```
void drawLine(int x1, int y1, int x2, int y2);

void drawRect(int x, int y, int width, int height);

void fillRect(int x, int y, int width, int height);

void drawOval(int x, int y, int width, int height);

void fillOval(int x, int y, int width, int height);
```

Color Class

- In order to work with colors in your GUI application you use the Color class.
- Commonly Used Constructor(s):
  - `Color(int r, int g, int b)`
  - `Color(float r, float g, float b)`
- Commonly Used Method(s):
  - `int getRed()`
  - `int getGreen()`
  - `int getBlue()`
  - `Color darker()`
  - `Color brighter()`
- Objects of class Color are immutable.

Color Class

- There are 13 predefined color objects in Java.
- They are all declared as **`public static final`** objects in class Color :
  - **`Color.RED`**
  - **`Color.ORANGE`**
  - **`Color.PINK`**
  - **`Color.YELLOW`**
  - **`Color.GREEN`**
  - **`Color.BLUE`**
  - **`Color.CYAN`**
  - **`Color.MAGENTA`**
  - **`Color.GRAY`**
  - **`Color.DARK_GRAY`**
  - **`Color.LIGHT_GRAY`**
  - **`Color.WHITE`**
  - **`Color.BLACK`**

Color Class

- To specify a certain color to be used when drawing on GUI Graphics object use the following method of class Graphics:

  - `void setColor (Color c)`
    - `g.setColor(Color.YELLOW)`

    - `this.setBackground(new Color(255,255,200)`

Font Class

- In order to create and specify fonts in your GUI application you use the Font class.
- Commonly Used Constructor(s):
  - `Font(String name, int style, int size)`
- To specify a certain font to be used when drawing on GUI Graphics object use the following method of class Graphics:
  - `void setFont (Font f)`
  - `g.setFont(new Font("Aril",Font.BOLD,4)`

Font Class

- To obtain the list of basic fonts supported by all platforms you can write the following line of code:

```
String[] s = Toolkit.getDefaultToolkit().getFontList();
```

- Objects of class Font are immutable.

Example

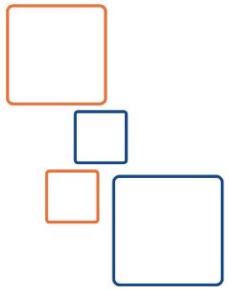
- Build a GUI program that shows all fonts and apply it

```
public class FontList extends JPanel {

    @Override
    public void paint(Graphics g) {
        String[] families = Toolkit.getDefaultToolkit().getFontList();

        for (int f = 0; f < families.length; f++) { // for each family

            Font font = new Font(families[f], style:Font.BOLD, size:18); // create font
            g.setFont(font); // set font
            String name = families[f];
            g.drawString( str:name, x:20, (f * 2 + 1) * 20); // display name

        }
    }
}
```

Course Outline

- **Day1**
  - Introduction to Java
  - Basic Java Concepts
- **Day2**
  - Data Types & Operators
  - using Arrays & Strings
  - Controlling Program Flow
  - Modifiers and Access Specifiers
- **Day3**
  - Simple GUI
  - Essential Java Classes
  - Java Exception
- **Day4**
  - Interfaces
  - Multi-Threading
- **Day5**
  - Inner class
  - Event Handling

Java Programming
Exceptions



Java™ Education
and Technology Services

Invest In Yourself ,
Develop Your Career

What are Exceptions?

- Exceptions are events that occur during the execution of programs that disrupt the normal flow of instructions (e.g. divide by zero, array access out of bound, etc.).
- In Java, an exception is an object that wraps an error event that occurred within a method and contains:
  - Information about the error including its type
  - The state of the program when the error occurred
  - Optionally, other custom information
- Exception objects can be thrown and caught.
- Exceptions are used to indicate many different types of error conditions.
- JVM Errors:
  - OutOfMemoryError ,StackOverflowError, LinkageError
- System errors:
  - FileNotFoundException, IOException, SocketTimeoutException
- Programming errors:
  - NullPointerException, ArrayIndexOutOfBoundsException, ArithmeticException

Checked vs. Unchecked Exceptions

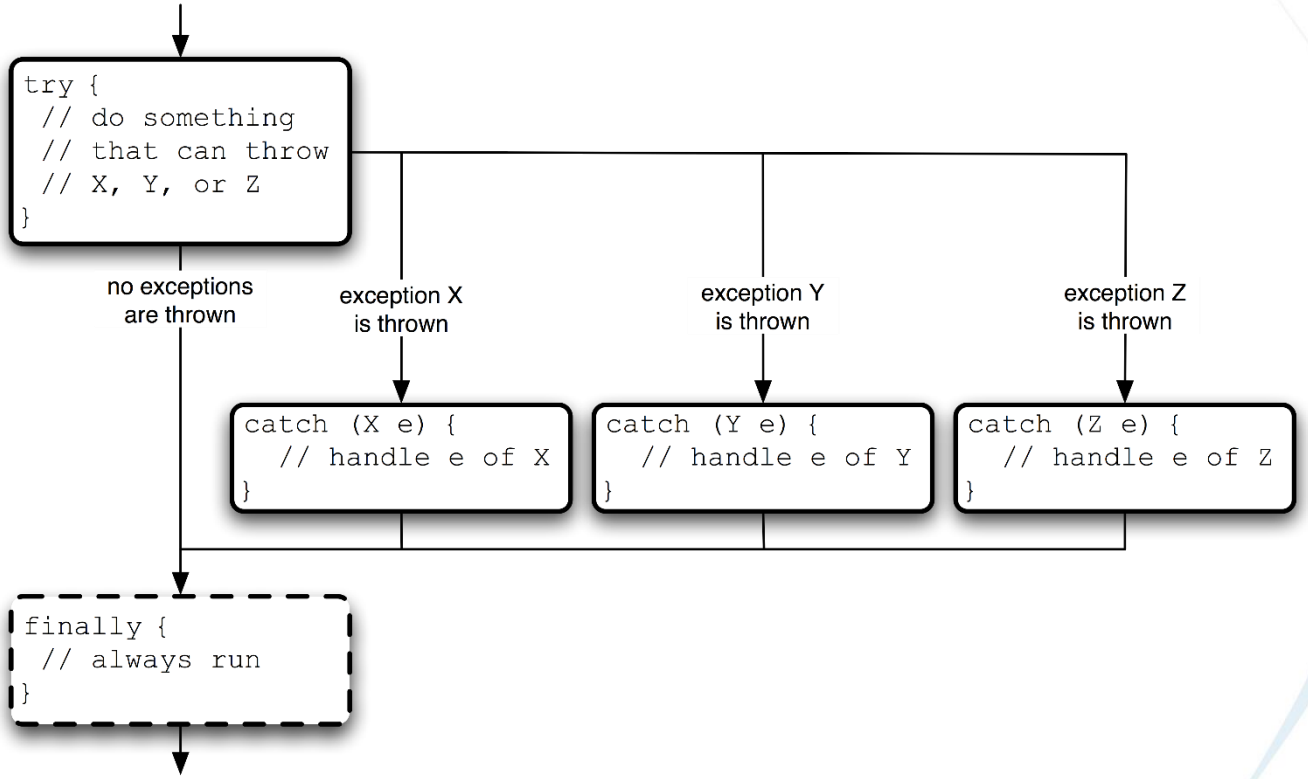
- Errors and RuntimeExceptions are unchecked – that is, the compiler does not enforce (check) that you handle them explicitly.
- Methods do not have to declare that they throw them (in the method signatures).
- It is assumed that the application cannot do anything to recover from these exceptions (at runtime).
- All other Exceptions are checked – that is, the compiler enforces that you handle them explicitly.
- Methods that generate checked exceptions must declare that they throw them.
- Methods that invoke other methods that throw checked exceptions should handle them (they can be reasonably expected to recover) .

Checked vs. Unchecked Exceptions

- Checked exceptions give API designers the power to force programmers to deal with the exceptions. API designers expect programmers to be able to reasonably recover from those exceptions, even if that just means logging the exceptions and/or returning error messages to the users.
- https://www.protechtraining.com/content/java_fundamentals_tutorial-exceptions

Handling Exceptions

```
try {
    // Code bock
}
catch (ExceptionType1 e1) {
    // Handle ExceptionType1 exceptions
}
catch (ExceptionType2 e2) {
    // Handle ExceptionType2 exceptions
}
// ...
finally {
    // Code always executed after the
    // try and any catch block
}
```



Handling Exceptions

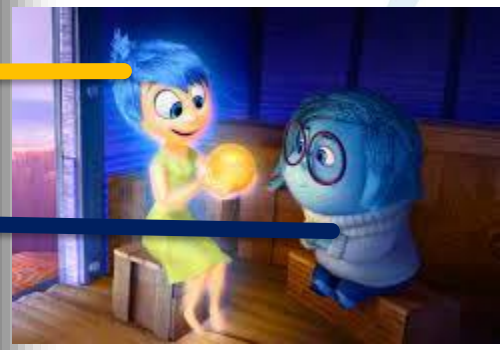
```
InputStreamReader inputStreamReader = new InputStreamReader(System.in);
BufferedReader reader = new BufferedReader(inputStreamReader);
System.out.println("Type name:");
```

unreported exception IOException; must be caught or declared to be thrown
---
(Alt-Enter shows hints)

```
    name = reader.readLine();
    System.out.println("Hello " + name);
```

```
public String readLine() throws IOException {
```

```
String name;

try {
    name = reader.readLine();
} catch (IOException ex) {
    Logger.getLogger(Lecture_Demo.class.getName()).log(Level.SEVERE, null, ex);
}
    System.out.println("Hello " + name);
```



Considerations Regarding Exceptions

- If several method calls throw different exceptions,
  - then you can do either of the following:
    1. Write separate `try-catch` blocks for each method.
    2. Put them all inside the same `try` block and then write multiple `catch` blocks for it (one `catch` for each exception type).
    3. Put them all inside the same `try` block and then just `catch` the parent of all exceptions: `Exception`.

Considerations Regarding Exceptions

- If more than one `catch` block are written after each other,
  - then you must take care not to handle a parent exception before a child exception
  - (i.e. a parent should not mask over a child).
  - Anyway, the compiler will give an error if you attempt to do so.

Example

```
try { ..... }
catch (FileNotFoundException e) {
  System.err.println("FileNotFoundException: “);
}
catch (IOException e) {
  System.err.println("Caught IOException: “);
}
```

Or:

```
try { ..... }
catch (FileNotFoundException | IOException e) {
  System.err.println(“.................“);
}
```

Java™ Education and Technology Services

Lab Exercise

Lab Exercise 1 – List of Fonts

- Create a GUI program that displays the list of available fonts in the underlying platform.
- Each font should be written in its own font.
- If you encounter any deprecated method|(s), follow the compiler instructions to re-compile and detect which method is deprecated. Afterwards, use the help (documentation) to see the proper replacement for the deprecated method(s).

Lab Exercise 2 – Drawing a Lamp

- Create a GUI program that makes use of the Graphics class drawing methods.
- You may draw the following lamp: